DOI 10.37882/2223-2966.2023.07.06

УПРАВЛЕНИЕ ЗАПАСАМИ НА ОС ANDROID И БАЗЫ ДАННЫХ FIREBASE С ИСПОЛЬЗОВАНИЕМ ШТРИХ КОДОВ И QR-КОДОВ

MANAGEMENT SYSTEM ON ANDROID OS AND FIREBASE DATABASES USING BARCODES AND QR CODES

Yu. Buzykova
A. Zufarova

Summary: We live in the era of the completion of the third digital revolution, which began in the second half of the last century. Its characteristic features are the development of information and communication technologies, automation and robotization of production processes.

The development of information technologies and means of communication, primarily electronic networks, creates a powerful impetus for the formation of a new trend in the functioning of modern business — the digitalization of economic relations. Most information carriers are becoming digital, which determines the main trend in the development of both modern technology and business processes with an overwhelming share of the electronic component. The electronic form of communication increases the level and efficiency of communication between buyers and sellers, creates new markets and opportunities for the reorganization of economic processes.

Automated inventory management system is the most important element of digitalization of the enterprise. And since a certain time, this is no longer an additional function, but a constant need for companies that want to compete in the market.

Keywords: programming languages, technologies, firebase, research.

Бузыкова Юлия Сергеевна
Доцент, МИРЭА — Российский технологический университет
juliaserg_buz@mail.ru
Зуфарова Анна Сергеевна
Старший преподаватель,
Тихоокеанский государственный университет
zoof_anna@mail.ru

Аннотация. Мы живем в эпоху завершения третьей цифровой революции, начавшейся во второй половине прошлого века. Ее характерные черты — развитие информационно-коммуникационных технологий, автоматизация и роботизация производственных процессов.

Развитие информационных технологий и средств коммуникации, прежде всего электронных сетей, создает мощный импульс для формирования новой тенденции функционирования современного бизнеса — диджитализации экономических отношений. Большинство носителей информации становятся цифровыми, что определяет основной тренд развития как современной техники, так и бизнес-процессов с подавляющей долей электронной составляющей. Электронная форма коммуникаций повышает уровень и эффективность общения между покупателями и продавцами и создает новые рынки и возможности для реорганизации экономических процессов.

Автоматизированная система управления запасами — важнейший элемент диджитализации предприятия. И с определенного времени это уже не дополнительная функция, а постоянная потребность компаний, которые хотят конкурировать на рынке.

Ключевые слова: языки программирования, технологии, Firebase, исследование.

Успешная диджитализация предусматривает комбинацию двух важнейших аспектов: перевод всех существенных активов в цифровой формат (например, областях только отсканированными документами, хранящимися в облаках) и использование специального программного обеспечения для диджитализации определенных процессов компании (например, использование SAP для выполнения различных транзакций с поставщиками без учета бумажного документооборота).

Цифровая трансформация — это серия общих изменений компании, что приводит к повышению эффективности и доходности компании [2]. Если вы не знаете, с чего начать автоматизацию вашего бизнеса, начните с разработки индивидуальной системы управления запасами. Такой инструмент, созданный непосредственно под вашу компанию, благодаря автоматизации, поможет мягко перейти на электронный документооборот, в то же время органично вписываясь в существующие процессы вашего предприятия. И помните — всегда принимайте решения на основе данных. А мы поможем вам эти данные собрать и обработать.

В дальнейшем такие системы будут интегрироваться с системами управления транспортными перевозками, системами автоматизации логистики запасов и закупок. Целью данной статьи является создание автоматизированной системы управления запасами на базе ОС Android для диджитализации товарооборота предприятий [5, с. 1934].

Именно через товарооборот происходит смена форм стоимости потребительских товаров, созданной в процессе производства. Традиционно товарооборот исследуется преимущественно на микроуровне как основной показатель деятельности торговых предприятий.

Однако на сегодня исследование товарооборота и факторов, которые на него влияют, должно происходить и на макроуровне, поскольку оптимальная структура товарооборота государства является одним из главных факторов конкурентоспособности ее экономики [10]. Запасы — это активы предприятия (ресурсы, имущество), которые:

— удерживаются для дальнейшей продажи (распределения, передачи) при условиях обычной хозяйственной деятельности;
— находятся в процессе производства с целью дальнейшей продажи продукта производства
— содержатся для потребления во время производства продукции, выполнения работ и оказания услуг, а также управления предприятием.

К запасам относятся, в частности:
— изготовленная на предприятии готовая продукция
— сельскохозяйственная продукция и продукция лесного хозяйства
— приобретенные (полученные) товары, содержащиеся предприятием с целью дальнейшей продажи
— сырье, основные и вспомогательные материалы, комплектующие изделия и другие материальные ценности, предназначенные для производства продукции, выполнения работ, оказания услуг, распределения, передачи, обслуживания производства и административных потребностей.

Увеличение запасов, с одной стороны, приводит к повышению эффективности, с другой — растет сумма средств, в частности, на хранение таких запасов. В то же время, сокращение запасов может привести к сбоям процессов производства, поставок или торговли. Поэтому важно иметь оперативную и актуальную информацию о запасах предприятия на складе по категориям, ценам и тому подобное. Со своей стороны использование программного обеспечения для управления запасами дает возможность всему бизнесу работать лучше. Так, в частности, вашим сотрудникам больше не потребуется регистрировать каждую единицу вручную, визуально проверять качество и количество полученного товара, ведь такой подход чреват риском ошибок человеческого фактора, а также требует больших временных затрат [4, с. 592].

По состоянию на данный момент существует несколько систем, которые предлагают готовые решения автоматизированных систем управления производством, в частности, Replenishment + от AbmCloud. Основная задача системы — обеспечить постоянное наличие сырья, материалов, комплектующих в нужном месте производственной цепочки в нужном количестве и в нужное время. А также сократить уровень избыточных запасов, обеспечить высокую надежность поставок и сократить влияние изменения спроса на уровень запасов. Это очень функционально мощный конкурент, но для управления такой системой нужны рабочие места и компьютеры [7]. Тогда как система, которую разрабатывает автор, нацелена больше на диджитализацию.

Такой системой можно управлять с мобильного девайса в любом уголке мира, сканировать товары только из телефона и добавлять в состав инвентаря.

1. Формирование требований к системе

Система должна давать возможность пользователю сканировать запасы по коду товара. Возможность добавлять выбранные товары к интвентарю склада по цене, названию, категории товара и штрих коду. Для каждой позиции товара пользователь может воспользоваться поиском по штрих коду, чтобы вывести информацию об изделии. Для каждого нового зарегистрированного аккаунта в системе предусмотрена база данных, которая хранит информацию о пользователе, а также об инвентаре в его системе. Система со своей стороны также должна автоматически подсчитывать количество единиц товара на складе и их ценовую составляющую. Как расшифровывать информацию, заложенную в QR-кодах?

Эту задачу успешно решает подавляющая часть современных смартфонов и планшетов. Для считывания закодированных данных достаточно перейти в окно приложения товара и активировать панель сканера, QR-код будет отсканирован встроенной камерой мобильного устройства.

Заказчики с поставщиками смогут общаться и обсуждать условия закупки новых запасов, не боясь излишков товара на складе, ведь вся информация будет указана в приложении.

Отвечает за выкладку товара, установку сопутствующего необходимого оборудования, размещает POS-материалы). Поскольку система встроена в мобильный девайс, то функционал дает возможность регистрировать новые товары, используя лишь телефон или планшет, искать изделия по номеру штрих кода [9]. Итак, в системе укомплектовано все необходимое в единое целое, в одну большую сеть, где можно хранить и вести подсчет товаров, не используя стационарных компьютеров. Система, ориентированная на потребности торговой компании в процессе автоматизации управления, предоставляет такие возможности, как реклама нового товара, просмотр собственных добавленных товаров к инвентарю, поиск товара по штрих коду, удаление товара, просмотр общего количества товаров и цен.

Во время установки APK в системе Андроид устанавливается файл DEX, который содержит код, ресурсы, скомпилированные как двоичные файлы. Файл dex обычно имеет тот же размер, что и файл apk, если у вас нет ресурсов, не скомпилированных в активах. Еще одной распространенной особенностью установки приложений является то, что android SAVES оригинальный apk при переустановке в случае ошибок или по каким-то дру-

гим причинам также устанавливается. Вот почему объем памяти наших приложений в системе вдвое больше.

Это сумма размера исходного apk и установленного dex. Проектирование базы данных начинается в коде AndroidStudio, где определяются типы данных для конкретного loyaute.xml файла в Common Attributes. Структура проецируется в классах, конкретный класс программы может отвечать за отдельное окно, такое как, например, регистрация пользователя, добавление продуктов, сохранение их в базе данных и тому подобное [3]. Подключение базы данных к проекту также является неотъемлемой частью. К тому же для пользования базы данных Firebase надо импортировать соответствующие библиотеки:

- import com.google.android.gms.tasks. OnCompleteListener
- import com.google.android.gms.tasks.Task
- import com.google.fi rebase.auth.AuthResult
- import com.google.fi rebase.auth. FirebaseAuth
- import com.google.fi rebase.auth. FirebaseUser
- import com.google.fi rebase.database. FirebaseDatabase

Разработанная структура базы данных включает создание переменных с соответствующими типами данных, а также разветвление на дочерние элементы.

Следующий фрагмент кода показывает добавление элемента в базу данных:

```
public void additem(){ String itemnameValue = itemname. getText().toString()
String itemcategoryValue = itemcategory. getText(). toString()
String itempriceValue = itemprice.getText(). toString()
String itembarcodeValue = itembarcode. getText(). toString()
fi nal FirebaseUser users = fi rebaseAuth. getCurrentUser()
String fi naluser=users.getEmail()
String resultemail = fi naluser. replace(“.”,”,”)
if (itembarcodeValue.isEmpty()) { itembarcode. setError(“Пусто “)
itembarcode.requestFocus()
return
} if(!TextUtils.isEmpty(itemnameValue)&&! TextUtils. isEmpty(itemcategoryValue)&&!Te xtUtils. isEmpty(itempriceValue)){ Items items = new Items(itemnameValue,it emcategoryValue,itempriceVal ue,itembarcod eValue)
databaseReference.child(resultemail). child(“Items”). child(itembarcodeValue). setValue(items)
databaseReferencecat. child(resultemail). child(“ItemByCatego ry”).child(itemcategoryValue). child(itembarcodeValue).setValue(items)
itemname.setText(“”)
itembarcode.setText(“”)
itemprice.setText(“”)
itembarcode.setText(“”)
Toast.makeText(additemActivity. this, itemnameValue” “добавлено”, Toast. LENGTH_SHORT).show()
}else{Toast.makeText(additemActivity.this,”пожалуйста, заполните все поля”,Toast.LENGTH_ SHORT).show ()
} }
```

таким образом разработан фрагмент кода добавления товаров в базу данных.

2. Реализация бизнес-логики

Бизнес-логика — это часть кода, которая выполняет логику приложения. Кроме бизнес-логики в приложениях может быть код, который отвечает за отображение информации, управление информацией, работу с внешними ресурсами/сервисами. В системе бизнес-логика сосредоточена в классах, которые синхронизированы с базой данных, хранящей информацию в формате обмена программирования данных JSON. Кроме этого, при запросах на модификацию данных передаются соответствующие заголовки авторизации во избежание несанкционированного доступа.

Ниже приведены несколько реализаций классов. Класс scanItemsActivity предназначен для работы со штрих-кодами и QR кодами, например, поиск товара по номеру кода. Класс образует два окна: одно-поиск товара по коду, а другое — вывод товара с параметрами, такими как: название изделия, его код, цена и категория. Также вызывается хранимая процедура, которая собирает из базы данных информацию и возвращает результат в формате JSON. Класс сервиса состоит из следующих методов:

1. метод fi rebasesearch. Метод создает запрос к базе данных, номера кода товара, который должен вернуть или вывести на экран, а также создает объект FirebaseRecyclerAdapter, который связывается с Query в RecyclerView. Когда данные добавляются, удаляются или изменяются, эти обновления автоматически применяются к интерфейсу пользователя в реальном времени

```
public void fi rebasesearch(String searchtext) { Query fi rebaseSearchQuery = mdatabaseReference. orderByChild(“itembarcode”). startAt(searchtext). endAt(searchtext+”\uf8ff ”)

FirebaseRecyclerAdapter fi rebaseRecyclerAdapter = new FirebaseRecyclerAdapter ( Items.class, R.layout.list_ layout, UsersViewHolder.class, fi rebaseSearchQuery ) {
```

2. метод setDetails. Метод создает TextView для вывода их на экран приложения после поиска конкретного товара, а именно код товара, название, категорию и цену. По аналогичной схеме реализован класс

(viewInventoryActivity) для работы с инвентарем склада, но выводятся все товары без поиска.

public void setDetails(Context ctx,String itembarcode, String itemcategory, String itemname, String itemprice){ TextView item_barcode = (TextView) mView.fi ndViewById(R. id.viewitembarcode)

TextView item_name = (TextView) mView.fi ndViewById(R. id.viewitemname)

TextView item_category = (TextView) mView.fi ndViewById(R. id.viewitemcategory)

TextView item_price = (TextView) mView.fi ndViewById(R. id.viewitemprice)

item_barcode. setText(itembarcode)

item_category. setText(itemcategory)

item_name.setText(itemname)

item_price.setText(itemprice)

} 3. Метод onCreate. Метод работает с базой данных, создает точку входа для доступа к базе данных Firebase. Возможность получить экземпляр, использовав getInstance(). Чтобы получить доступ к местоположению в базе данных и прочитать или записать данные, создана getReference().

@Override protected void onCreate(Bundle savedInstanceState) { super. onCreate(savedInstanceState)
setContentView(R.layout. activity_scan_items)
fi rebaseAuth = FirebaseAuth. getInstance()
fi nal FirebaseUser users = fi rebaseAuth.getCurrentUser()
String fi naluser=users. getEmail()
String resultemail = fi naluser. replace(“.”,””)
mdatabaseReference = FirebaseDatabase.getInstance(). getReference(“Users”). child(resultemail).child(“Items”)
resultsearcheview = fi ndViewById(R.id.searchfi eld)
scantosearch = fi ndViewById(R. id.imageButtonsearch)
searchbtn = fi ndViewById(R. id.searchbtnn)
mrecyclerview = fi ndViewById(R.id.recyclerViews)
LinearLayoutManager manager = new LinearLayoutManager(this)
mrecyclerview. setLayoutManager(manager)
mrecyclerview. setHasFixedSize(true)
mrecyclerview. setLayoutManager(new LinearLayoutManager(this))
}

Класс RegisterActivity предназначен для работы с пользователями системы управления запасами, регистрации их аккаунтов и сохранение в базе данных, чтобы хранить только ту информацию о товаре, которую добавлял конкретный пользователь. Конфиденциальность данных в базе присутствует, поэтому пароли не выводятся для безопасности данных. Класс состоит из следующих методов: 1. метод onCreate.

Содержит логику регистрации пользователя, создает текстовые поля для заполнения данных, таких как емейл, пароль, повтор пароля и имя . Чтобы получить объект FirebaseAuth, вызывает статический метод getInstance (). @Override protected void onCreate(Bundle savedInstanceState) { super. onCreate(savedInstanceState)
setContentView(R.layout. activity_register)
editTextName = fi ndViewById(R.id.departmentName)
editTextEmail = fi ndViewById(R.id.emailRegister)
editTextPassword=fi ndViewById(R.id.passwordRegister)
editTextcPassword= fi ndViewById(R.id.confi rmPassword)
UserRegisterBtn= fi ndViewById(R.id.button_register)
progressBar = fi ndViewById(R.id.progressbar)
progressBar. setVisibility(View.GONE)
mAuth = FirebaseAuth. getInstance()
UserRegisterBtn. setOnClickListener(new View. OnClickListener() { @Override public void onClick(View v) { registerUser()
} })
}

2. метод onStart. Обрабатывает уже зарегистрированного пользователя.

3. метод registerUser. Обрабатывает поля, бывшие в методе onCreate, предоставляет им тип данных .toString(), если поля неправильно заполнены или пустые выдает соответствующее сообщение. Метод получает данные и отправляет данные на основе электронной почты пользователя в базу данных, а также выводит сообщения в случае успешной или неуспешной регистрации. Класс additemActivity нужен для добавления товара в инвентарь склада. Также он создает поля для ввода данных о выбор, таких как: название товара, категория, цена и числовой тип данных для ввода номера кода товара. Кроме того, класс от — 19 инструментальные средства и среды программирования следует изменение в полях во время хранения товара и добавляет к базе данных [8, с. 1654]. Класс состоит из следующих методов:

1. метод onCreate. Содержит логику для регистрации товара, создает текстовые поля для заполнения данных, таких как: название товара, категория, цена и числовой тип данных для ввода номера кода товара. Чтобы получить объект FirebaseAuth, вызывается статический метод getInstance (). Ниже приведена реализация: @Override protected void onCreate(Bundle savedInstanceState) { super. onCreate(savedInstanceState)
setContentView(R.layout. activity_additem)
fi rebaseAuth = FirebaseAuth. getInstance()

databaseReference = FirebaseDatabase.getInstance(). getReference("Users")
databaseReferencecat=FirebaseDatabase.getInstance(). getReference("Users")
resulttextview = fi ndViewById(R.id.barcodeview)
additemtodatabase = fi ndViewById(R. id.additembuttontodatabase)
scanbutton = fi ndViewById(R.id.buttonscan)
itemname = fi ndViewById(R. id.edititemname)
itemcategory= fi ndViewById(R.id.editcategory)
itemprice = fi ndViewById(R. id.editprice)
itembarcode= fi ndViewById(R.id.barcodeview)
scanbutton. setOnClickListener(new View. OnClickListener() { @Override public void onClick(View view) { startActivity(new Intent(getApplicationContext(), ScanCodeActivity.class))
} })
additemtodatabase. setOnClickListener(new View. OnClickListener() { @Override public void onClick(View v) { additem()
} })
}

2. метод additem. Выполняет такие функции, как: добавление элемента в базу данных, создание полей для их заполнения, передача этих полей в базу данных, хранение.

3. метод Logout. Нужен для вызова меню в верхней части экрана и выхода из аккаунта. Этот метод фигурирует также в других классах, где есть возможность вызова меню. Приведена реализация ниже: private void Logout () { fi rebaseAuth.signOut()
fi nish()
startActivity(new Intent(additemActivity. this,LoginActivity.class))
Toast.makeText(additemActivity. this,"LOGOUT SUCCESSFUL", Toast.LENGTH_SHORT).show()
}

4. Метод onCreateOptionsMenu. Подобен для MenuInflater-это системный ресурс Android. Создается во время загрузки андроида. Это постоянный объект, и ссылка на него всегда доступна в памяти. Каждый подкласс класса Context, то есть Activity может получить ссылку на него, вызвав getMenuInfl ater()изнутри класса. Приведена ниже реализация: 20 инструментальные средства и среды программирования
@Override public boolean onCreateOptionsMenu (Menu menu) { getMenuInfl ater ().infl ate(R.menu. menu,menu)
return true
} Класс ScanCodeActivity.

Отвечает за сканирование штрих кодов и QR кодов. Ниже приведена реализация (использована библиотека ZXingScannerView scannerView):

import com.google.zxing. Result
import me.dm7. barcodescanner.zxing. ZXingScannerView
public class ScanCodeActivity extends AppCompatActivity implements ZXingScannerView. ResultHandler { int MY_PERMISSIONS_ REQUEST_ CAMERA=0
ZXingScannerView scannerView
@Override protected void onCreate(Bundle savedInstanceState) { super. onCreate(savedInstanceState)
scannerView = new ZXingScannerView(this)
setContentView(scannerView)
} @Override public void handleResult(Result result) { additemActivity. resulttextview.setText(result. getText())
onBackPressed()
} @Override protected void onPause() { super.onPause()
scannerView.stopCamera()
} @Override protected void onPostResume() { super. onPostResume()
if (ContextCompat.checkSelf Permission(getApp licationContext(), Manifest.permission.CAMERA) != PackageManager. PERMISSION_GRANTED) { ActivityCompat. requestPermissions(this, new String[] {Manifest.permission. CAMERA}, MY_PERMISSIONS_ REQUEST_CAMERA)
} scannerView. setResultHandler(this)
scannerView.startCamera()
}}

Таким образом была реализована панель сканирования для штрих кодов и QR кодов.

3. Тестирование

В структуре Android Studio есть папки androidTest и test рядом с основной папкой с классами проекта. Зачем нужно тестирование? Если приложение маленькое, то тесты не нужны и вполне можно обходиться без тестирования и дальше. Почему? Речь идет о том, что в небольших проектах можно контролировать логику программы.

Также можно предсказать слабые места и исправить код. Но все меняется, если программа стала сложной. Если появилось более десятка различных экранов активностей, отдельных классов и тому подобное, код следует разбивать на модули, чтобы обеспечить независимость. Такой подход обязательно используется в компаниях, где каждый отвечает за свой участок кода [6]. Тесты делятся на две категории — локальные (Unit Testing) и инструментальные (UI Testing).

Инструментальные средства и среды программирования Локальные тесты проверяют работу метода, класса, компонента. Тест не зависит от Android. Практично, проверяется код Java, который можно контролировать на обычном компьютере без участия устройства или

эмулятора. Например, такому варианту соответствует сложение двух чисел типа int. Подобные тесты проводят в папке Test.

Для инструментальных тестов наличие устройства или эмулятора является обязательным, поскольку нужно тестировать нажатие кнопки, ввод текста, прокрутки, касания и другие операции. Тесты проводятся в папке androidTest. Выводы Реализовано программное средство для автоматизации запасами на основе ОС Android с использованием базы данных Firebase и языка программирования Java. Система использует метод считывания штрих кодов и QR-кодов.

Он обеспечивает добавление товара на склад со всеми подробностями о выбранном товаре и просмотр всех запасов на складе и их цене, категории, названию и коду товара. Реализована авторизация пользователей системы с помощью открытого стандарта FirebaseAuth и FirebaseDatabase.

Предоставлена возможность регистрировать пользователей системы, что позволило сделать систему масштабируемой. В системе реализован поиск товара по коду, а также просмотр запасов на складе с автоматическим подсчетом цены [1]. Основной сервис базы данных-облачная СУБД класса NoSQL, позволяющая сохранять и синхронизировать данные между несколькими клиентами. Предусмотрен API для шифрования данных. Система в денежном эквиваленте автоматически рассчитывает результаты управления запасами товаров и общую цену. Библиотека me.dm7.barcodescanner.zxing.ZXingScannerView использовалась для реализации сканера. Предложенная автоматизированная система управления запасами делает возможным избрание наиболее эффективных инструментов для регулирования товарооборота и управления запасами.

ЛИТЕРАТУРА

1. Александровский В.Г. Мобильные технологии в строительстве. Программное обеспечение на платформе Android. Часть 1 // Инженерный вестник Дона, 2019, №4. URL: ivdon.ru/ru/magazine/archive/n4y2019/5874
2. Документация для разработчика Firebase [Электронный ресурс] — Режим доступа https://firebase.google.com/docs
3. Нодира Махкампўлатовна Алиева, Гулчехра Алиаскаровна Кадирова, Саида Искандаровна Джураева РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ИЗУЧЕНИЯ ЯЗЫКОВ В СРЕДЕ ANDROID // Academic research in educational sciences. 2021. №11. URL: https://cyberleninka.ru/article/n/razrabotka-informatsionnoy-sistemy-izucheniya-yazykov-v-srede-android
4. Харди Б., Филлипс Б. Программирование под Android. Для профессионалов. — СПб.: Питер, 2014. — 592 с.: ил.
5. Abdukarimov S., Shokirov R. Team competitions on programming format ACM ICPC // International journal of scientific and technology research. 2020. № 4(9). С. 1932–1935.
6. Broyde L., Nixon K., Chen X., Li H., Chen Y. MobiCore: An adaptive hybrid approach for power-efficient CPU management on Android devices. // Proc. of 30th IEEE International System-on-Chip Conference, SOCC, 2017.
7. Chen Y.L., Chang M.F., Yu C.W., Chen X.Z., Liang W.Y. Learning-directed dynamic voltage and frequency scaling scheme with adjustable performance for single-core and multi-core embedded and mobile systems. // Sensors. 2018. Vol. 12, N 9.
8. Granichin O.N., Amelina N. Simultaneous perturbation stochastic approximation for tracking under unknown but bounded disturbances. // IEEE Trans. on Automatic Control. 2015. Vol. 60, N 6. P. 1653–1658.
9. Mentsiev A.U., Alams M.T. Mobile forensic tools and techniques: Android data security // Инженерный вестник Дона, 2019, №2. URL: ivdon.ru/ru/magazine/archive/n2y2019/5766
10. Poornambigai K., Raj M. L., Meena P. Reducing the energy consumption using DVFS performance optimizing scheme. // EPRA International Journal of Research and Development (IJRD). 2017. Vol. 2, N1.

Журнал «Современная наука: актуальные проблемы теории и практики»